

Compile with MinGW

Note

These instructions pre-date OpenCPN v4.2 because they suggest using wxWidgets 2.8.12! The process for a more modern mingw compile is probably very similar, but the versions would be different. Circa 2010-2012

MinGW is Minimalist GNU for windows. There is a free compiler (gcc) which may be used to compile OpenCPN to run on windows. It can be installed from windows, or used as a cross compiler under linux. The run-time performance of OpenCPN is identical to the visual studio compiled version, the compiler produces more informative error messages, and uses less disk space. You will also find the amount of manual copying of files to be much less.

Important note before you start

The order of the steps described below really matters. Don't skip any steps not explicitly marked as optional and don't change their order unless you really know what you are doing. It's an excellent idea to read the whole text first and make sure you understand what it's talking about, especially if you are new to software development.

If you encounter any problems, please get to us in the forum. Cruisers Forum Thread is here: [building-opencpn-using-mingw-gcc-on-windows-121802.html#post1600285](http://cruisersforum.com/forums/thread-121802.html#post1600285) Make sure you mention you are using MinGW not Visual Studio.

Preparing the toolchain

Note about LLVM

No one has used LLVM with MinGW runtimes to build OpenCPN. In theory it will “just work” but until then. it is completely experimental. Until then, gcc is the proven option.

MinGW

Get it from <http://sourceforge.net/projects/mingw/files/> Run the setup program and be sure to select mingw-developer-toolkit (first in the list) as well as mingw32-gcc-g++. If you have already installed gettext for compiling with visual studio, you are all set, otherwise scroll down and select the msys-gettext binary packages. The following tools are also needed for compiling with visual studio, so if you already have them installed, skip past these.

Git for Windows

Get it from <https://git-for-windows.github.io/> (You want Git for Windows or MSysGit)

The defaults for all the installation settings are fine except the following:

* On “Configuring the line ending conversions” select “Checkout as-is, commit Unix-style line endings” (Git 2.11.0)

- this is really important, the codebase uses Unix Lf line endings and committing Cr/Lf makes the commits huge and hides their real content. If your git is already installed, please edit the **.git/config** in your local working copy and set **autocrlf = input** in the **[core]** section on top.

If you want a tutorial, have a look at the series of articles starting at

http://www.lostechies.com/blogs/jason_meridth/archive/2009/06/01/git-for-windows-developers-git-series-part-1.aspx

You may already have git installed for compiling under Microsoft MSVC++ and that will be sufficient.

Cmake

Get it from <http://cmake.org/cmake/resources/software.html>

While installing it's advisable to let the setup program add cmake to the system PATH variable

You may already have cmake installed for compiling under Microsoft MSVC++ .

NSIS (Unicode)

Needed **only** in case you intend to build the **installation package** for OpenCPN. Not needed for normal development.

Get it from <http://code.google.com/p/unsis/downloads/list>

It is recommended to use NSIS 3.00b or newer, and NSIS 3.00b is required to make install packages. This program must be installed into an “NSIS” directory or path names will not be right and program's will not be found.

[You cannot do a parallel install into a different directory such as NSIS3.0b if you have already installed NSIS2.46 for compiling with Microsoft Visual Studio. Therefore if you have been compiling Microsoft MSVC++ you must uninstall NSIS 2.46 and then install NSIS 3.0b to the NSIS directory. Then you can compile MinGW install packages. Then when you are going to compile in Microsoft Visual Studio, reverse the process and uninstall NSIS3.0b and install NSIS2.46 to the NSIS directory, otherwise you will have problems making the packages. There is the option of tweaking the registry keys below, which I have not confirmed yet, (Rick)]

Note these instructions are copied from the Visual Studio page (and should be modified for NSIS 3.0b):

1. The NSIS Unicode 2.46 setup file is also included in the GIT repository →
..\buildwin\NSIS_Unicode\nsis-2.46-Unicode-setup.exe
2. Direct download link:
<http://code.google.com/p/unsis/downloads/detail?name=nsis-2.46.3-Unicode...>
3. Download and install it. Due to a “bug” in CMake, which only looks at
“HKEY_LOCAL_MACHINE\SOFTWARE\NSIS” for the installation location of NSIS and the Unicode
version adds its registry key in “HKEY_LOCAL_MACHINE\SOFTWARE\NSIS\Unicode”, there is some
registry tweaking needed.
4. Just copy the value (it's the installation path) of the
“HKEY_LOCAL_MACHINE\SOFTWARE\NSIS\Unicode” key to
“HKEY_LOCAL_MACHINE\SOFTWARE\NSIS”.
5. Alternatively you can just run the batch file 'CopyNSISUnicodeRegKey.bat' which is also included in
the GIT repository → ..\buildwin\NSIS_Unicode\CopyNSISUnicodeRegKey.bat Depending on your
security settings, mainly on Windows 7 and newer, you may have to run it as Administrator (right-
click the file and select “Run as administrator.”)
6. This means if you also want to use the ANSI NSIS version you first have to change the value of
“HKEY_LOCAL_MACHINE\SOFTWARE\NSIS” registry key according to the installation path of the
respective version you want to use.

To make the installer package use proper language name translations, it's necessary to modify file
X:\Program Files\NSIS\Unicode\Contrib\Language files\Norwegian.nsh and change the line

```
!insertmacro LANGFILE “Norwegian” “Norwegian”
```

to

```
!insertmacro LANGFILE “Norwegian” “Norsk”
```

Pre-requisites

WxWidgets

Download wxMSW-2.8.12-Setup.exe from http://www.wxwidgets.org/downloads/#latest_stable and install
it. This version is tested, and newer versions are not guaranteed to work without errors. [You may
already be using this version if you are compiling under Microsoft Visual Studio. If so you must compile
separately for mingw, creating a separate directory “build-mingw”, compiling with MSYS and then use
those binary files.]

Necessary modifications for MinGW

Go to the folder where you installed/unpacked wxWidgets and edit the file **src/msw/treectrl.cpp** to add
the lines in red starting from line 43:

```
+++ treectrl.cpp 2014-02-28 11:27:59.367810324
```

```
+0800 @@ -43,6 +43,10 @@  
#include "wx/msw/dragimag.h"  
#include "wx/msw/uxtheme.h"
```

```
#ifdef MINGW32  
#define TV_DISPINFO NMTVDISPINFO  
#endif
```

macros to hide the cast ugliness

Compile wxWidgets from Msys shell

Modify the user Environment PATH by adding "c:/mingw/bin;" Win7 computers do not require reboot.
Computer > Right Click > Properties > Advanced System Settings > Environment Variables
Have now set these additional paths:

```
;C:\MinGW\bin;C:\MinGW\MSYS\1.0\local\bin;C:\MinGW\MSYS\1.0\bin
```

Start up the msys shell (run msys.bat located by default in C:\mingw\msys\1.0\msys.bat

To execute the MSYS.bat file located in directory C:\MinGW\msys\1.0
Run CMD window and cd c:\MinGW\msys\1.0, then execute "msys" you will then be in the MSYC command window.

When in MSYS the phrasing for commands is a little different than Microsoft's compiler.

Example: \$ cd /c/mingw takes me to c:/mingw

-To compile wxWidgets I found "\$ cd \${WXWIN}" took me to /c/wxwidgets-2.8.12" because my environment variable is defined as {WXWIN}.

Compile wxWidgets from the MSYS shell

```
cd /c/${WXDIR} //maybe use "WXWIN"  
mkdir build-mingw  
cd build-mingw  
../configure --with-opengl --enable-unicode  
make install
```

-This should run for a very very long time. If it ends with "Compiler not found" you must set the PATH above.

Near end "if test ! -d /usr/local/include/wx-2.8/ 'dirname \$f ; \fi; \ /bin/install -c m 644 ../include/\$f /usr/local/include/wx-2.8/\$f; \ Then other lines show "No such file or directory"
Finally ends with "The installation of wxWidgets is finished..."

Compile Debug

Pass `-enable-debug` to configure to enable debugging support and to compile without optimizations. EG: `"../configure --with-opengl --enable-unicode --enable-debug"` Then "make install" (as above.)

Building zlib

zlib is needed to build the grib plugin: Download `zlib-1.2.8.tar.gz` from <http://www.zlib.net>

```
tar xavf zlib-1.2.8.tar.gz
cd zlib-1.2.8
make -f win32/Makefile.gcc
cp libz.a /usr/local/lib
```

I downloaded to the download directory, used Jzip to expand it into a new directory under downloads, then used MSYS.bat to compile it. Then browse to the downloads `C:\Users\...\Downloads\zlib-1.2.8` Find and copy "libz.a" to from `C:\Users\...\Downloads\zlib-1.2.8` to `C:\MinGW\lib\gcc\mingw32\4.8.1`

Getting the OpenCPN source

From the msys shell, cd into the directory you wish to put the opencpn source code. To get the source, (note the MinGW specific changes are not merged to the main tree) for the first time, issue

```
git clone git://github.com/seandepagnier/OpenCPN.git -b mingw //
```

/To update the code you cloned before, cd into the source directory and issue `git pull` The wxWidgets mingw libraries are not in the git repository. Copy wxWidgets libraries into OpenCPN/wxWidgets-mingw:

```
mkdir wxWidgets-mingw
cp -r /c/wxWidgets-2.8.12/build-mingw/lib/*dll wxWidgets-mingw
```

Note that the correct directory is `C:\...\[opencpn-mingw]\wxWidgets-mingw` if you are going to be distributing a package, please remove the wxWidgets libraries that we do not use.

Building the OpenCPN source

Create a directory named **build** under the topmost source directory (Because I had been compilling using Microsoft MSVC++ and the build directory was full of Microsoft, I deleted the "build" directory and all its contents because it was created by the MSVC++ compiler. Then I mkdir "build" again.)

Configuring alternative 1: From the command line (recommended):

cd into the **build** directory Using MSYS.bat

```
$ cd /c/Data-Dart/Up-Soft/Navigation/opencpn-mingw
```

\$ cd build Issue: By default OpenCPN is installed into C:\Program Files\OpenCPN Unfortunately the make program is not allowed to copy files there. To build and install to an alternate location:

```
cmake -G "MSYS Makefiles" -DCMAKE_INSTALL_PREFIX=. .. <--detects, checks, finds, stages, build, stage, writes, config, generate, written  
make <--List and build to 100%  
make install <--builds and links.  
./opencpn <--Program should open and operate.
```

If building a package and will use the installer program to copy to Program Files:

```
cmake -G "MSYS Makefiles" .. <--Stages, configures, generates  
make <-- Builds targets  
make package <-- Builds targets for the package, does cpack  
./opencpn_* <-- Executes and installs the package.
```

After “make package” if you get errors regarding LZMA make sure you have installed NSIS 3.0b into the NSIS directory. It is required.

Debugging:

Simply change into the directory with opencpn (in program files) and execute:

```
gdb opencpn
```

From the (gdb) prompt, type “run” or any other gdb command.

From:
<https://opencpn.org/wiki/dokuwiki/> - OpenCPN Manuals

Permanent link:
https://opencpn.org/wiki/dokuwiki/doku.php?id=opencpn:developer_manual:developer_guide:compiling_windows:compiling_windows_mingw

Last update: 2020/03/07 15:13

